Statistical and Mathematical Methods for Artificial Intelligence

Matteo Donati

October 22, 2021

Contents

Chapter 1

Numerical Computation and Finite Numbers

A generic problem solving process can be summarized in the following steps:

1. Development of a mathematical model.
2. Development of algorithms for the relative numerical solution.
3. Implementation of such algorithms in computer software.
4. Execution of such software to simulate the physical process numerically.
5. Graphical visualization of the computed results.
6. Interpretation and validation of such results.

The various sources of approximation of this process are the following:

- **Measure errors**, due to the measure instrument.
- **Arithmetic errors**, due to the propagation of the rounding errors of each single operation in an algorithmic process.
- **Truncation errors**, due to the truncation of an infinite procedure to a finite procedure.
- **Inherent errors**, due to the finite representation of the data of a problem.

1.1 Numerical Computation

In particular, below are some useful definitions used in numerical computation:

- Let $\tilde{x}$ be an approximation of a given number x, then the **absolute error** of such approximation is the quantity:

$$E_x = |\tilde{x} - x| \tag{1.1}$$

  However, this quantity could be meaningless in some applications.

- The **relative error** of such approximation is defined by the quantity:

$$R_x = \left| \frac{\tilde{x} - x}{x} \right|, \ x \neq 0 \tag{1.2}$$

- The **precision** is the quantity which reflects the number of digits with which a number is expressed.

- The **accuracy** is the number of correct significant digits used for approximating some quantity

- The number $\tilde{x}$ is said to approximate x to d significant digits if d is the largest non-negative integer such that:

$$\left| \frac{\tilde{x} - x}{x} \right| \leq \frac{10^{1-d}}{2} \tag{1.3}$$

1.2 Finite Numbers

In particular, below are some useful definitions used in finite numbers representation:

- Given a base $\beta \in \mathbb{Z}$, with $\beta > 1$, a number $x \in \mathbb{R}$, with $x \neq 0$, can be expressed in a unique way as:

$$\begin{aligned} x &= \text{sign}(x)(d_1\beta^{-1} + d_2\beta^{-2} + \dots)\beta^p \\ &= \text{sign}(x)m\beta^p \end{aligned} \tag{1.4}$$

  where:

  - $p \in \mathbb{Z}$.
  - The digits $d_1, d_2, \dots$ satisfy the following conditions:
    - $0 \leq d_i \leq \beta - 1$.
    - $d_1 \neq 0$ and d_i are not all equal to $\beta - 1$ from a certain index i onward.
  - $\frac{1}{\beta} \leq m < 1$ is the **mantissa**.
  - β^p is the **exponential part**.

- The **normalized scientific representation** of any real number x is the following:

$$x = \text{sign}(x)(0.d_1 d_2 \dots)\beta^p \tag{1.5}$$

- Formally, a system of floating point numbers $\mathcal{F}(\beta, t, L, U)$ is defined by the following parameters:
  - The base β.
  - The precision t.
  - The exponent range $[L, U]$.

  Any floating point number $x \in \mathcal{F}$ has the following form:

$$x = \text{sign}(x)(d_1\beta^{-1} + \cdots + d_t\beta^{-t})\beta^p,\ L \leq p \leq U \tag{1.6}$$

  where $d_i \in \mathbb{Z}$ and $0 \leq d_i \leq \beta - 1$ with $i = 1, \dots, t$.

- A floating point system is normalized when $d_1 \neq 0$. In this case:

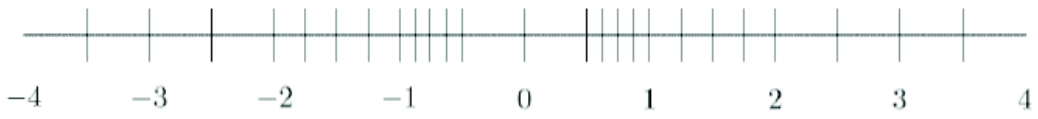
$$\frac{1}{\beta} \leq d_1 < 1 \tag{1.7}$$

  Normalization allows one to have a unique representation of each number and, assuming $\beta = 2$ (i.e. binary system), to not store the leading bit since it will always be 1.

- A floating-point number system is finite and discrete. In particular, the total number of normalized floating-point numbers is:

$$2(\beta - 1)\beta^{t-1}(U - L + 1) + 1 \tag{1.8}$$

  Moreover, the smallest positive normalized number is $UFL = \beta^L$ and the largest floating-point number is $OFL = \beta^{U+1}(1 - \beta^{-t})$. Floating-point numbers are equally spaced only between successive powers of β. For example, considering the set $\mathcal{F}(2, 3, -1, 1)$:



  the number of elements is: $2(2-1)2^2(1+1+1)+1 = 25$, $UFL = 2^{-1}$ and $OFL = 2^2(1-2^{-3}) = 3.5$.

- Not all real numbers are exactly representable. Rounding rules refer to the approximation of real numbers x to the floating point numbers $fl(x) \in \mathcal{F}$. In particular, the two most common rounding rules are the following:

  - **chop** rounding rule, which consists in truncating x after the t-th digit.
  - **round-to-nearest** rounding rule, which consists in rounding x to the nearest floating-point number.

- The accuracy of a floating-point system is characterized by the **machine precision**, ϵ_{mach}:

  - With chop rounding: $\epsilon_{mach} = \beta^{1-t}$.
  - With round-to-nearest rounding: $\epsilon_{mach} = \frac{1}{2}\beta^{1-t}$.

  Moreover, the maximum relative error in representing a real number x within the range of a certain floating-point system is given by:

$$\left| \frac{fl(x) - x}{x} \right| \leq \epsilon_{mach} \tag{1.9}$$

  This means that, during the execution of an algorithm, each calculation can produce an error which is at most equal to ϵ_{mach}.

- IEEE floating-point standard provides special values to indicate two exceptional situations:

  - `Inf`, which is the result of the division between a finite number and zero.
  - `NaN`, which is the result of undefined or indeterminate operations, such as $0/0$, $0 \cdot$ `Inf`, `Inf`/`Inf`.

- The result of a floating-point arithmetic operation may differ from the result of the corresponding real arithmetic operation on the same operands:

  - In additions and subtractions, the mantissa of one of the two operands is shifted in order to match the two exponents. This may cause the loss of some digits of the smaller number, possibly all of them.
  - In multiplication, the result of two t-digits mantissas contains up to $2t$ digits, so such result may not be representable.
  - In divisions, the quotient of two t-digits mantissas may contain more than t digits, such as the non-terminating binary expansion of $1/10$.

  Moreover, the real result may also fail to be representable because of its exponent being outside of the available range (i.e. $p > U$, which is called **overflow**, or $p < L$, which is called **underflow**).

- The subtraction between two t-digit numbers having the same sign and similar magnitudes yields a result having less than t digits. The reason is that the leading digits of the two numbers cancel each other. This is called **cancellation**. For example, considering $x = 1.92403 \cdot 10^2$ and $y = 1.92275 \cdot 10^2$:

$$fl(x) = 0.192403 \cdot 10^3,\ fl(y) = 0.192275 \cdot 10^3$$
$$\Rightarrow z = fl(x - y) = (0.192403 - 0.192275) \cdot 10^3 = 0.000128 \cdot 10^3$$
$$\Rightarrow fl(z) = 0.128000 \cdot 10^3$$

Chapter 2

Linear Algebra

2.1 Vector Spaces

A **vector space** V over a field F ($F = \mathbb{R}$ or $F = \mathbb{C}$) is a set closed under finite vector addition and scalar multiplication. The elements of V are called **vectors** while the elements of F are called **scalars**. In particular, vector addition and scalar multiplication must satisfy the following properties:

- Commutativity of addition:

$$\forall \boldsymbol{v}, \boldsymbol{w} \in V, \boldsymbol{v} + \boldsymbol{w} = \boldsymbol{w} + \boldsymbol{v} \tag{2.1}$$

- Associativity of addition:

$$\forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V, \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w} \tag{2.2}$$

- Existence of the identity element of addition, $\mathbf{0} \in V$, such that:

$$\forall \boldsymbol{v} \in V, \boldsymbol{v} + \mathbf{0} = \boldsymbol{v} \tag{2.3}$$

- Existence of the additive inverse:

$$\forall \boldsymbol{v} \in V, \exists - \boldsymbol{v} \in V, \boldsymbol{v} + (-\boldsymbol{v}) = \mathbf{0} \tag{2.4}$$

- Existence of the identity element of scalar multiplication, $1 \in F$, such that:

$$\forall \boldsymbol{v} \in V, 1\boldsymbol{v} = \boldsymbol{v} \tag{2.5}$$

- Compatibility of scalar multiplication with field multiplication:

$$\forall a, b \in F, \forall \boldsymbol{v} \in V, (ab)\boldsymbol{v} = a(b\boldsymbol{v}) \tag{2.6}$$

- Distributivity of scalar multiplication with respect to field addition:

$$\forall a, b \in F, \forall \boldsymbol{v} \in V, (a+b)\boldsymbol{v} = a\boldsymbol{v} + b\boldsymbol{v} \tag{2.7}$$

- Distributivity of scalar multiplication with respect to vector addition:

$$\forall a \in F, \forall \boldsymbol{v}, \boldsymbol{w} \in V, a(\boldsymbol{v} + \boldsymbol{w}) = a\boldsymbol{v} + a\boldsymbol{w} \tag{2.8}$$

Moreover, given a vector space V over the field F:

- The set W is a **subspace** of V if and only if:
  1. $W \subset V$.
  2. W is a vector space over F.
- The set W of all finite linear combinations of vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}, \boldsymbol{v}_i \in V$, is called **subspace spanned by** $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ and it is expressed as:

$$W = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\} = \left\{ \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i \middle| \boldsymbol{v}_i \in V, \alpha_i \in F \right\} \tag{2.9}$$

  In particular, the system $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ is called a **system of generators** for V.
- A system of vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}, \boldsymbol{v}_i \in V$ is called **linearly independent** if:

$$\alpha_1 \boldsymbol{v}_1 + \cdots + \alpha_m \boldsymbol{v}_m = \boldsymbol{0} \Rightarrow \alpha_1 = \cdots = \alpha_m = 0 \tag{2.10}$$

  with $\alpha_1, \ldots, \alpha_m \in F$. From a geometrical point of view, n vectors are linearly dependent if they lit on the same $(n-1)$-dimensional hyperplane.
- Any system of linearly independent generators of V is a **basis** for the specific vector space V.
  - If V is a vector space which admits a basis of n vectors, then every system of linearly independent vectors has at most n elements and any other basis of V has exactly n elements. The number n is called **dimension** of V ($\dim(V) = n$).

2.2 Matrices

Let m and n be two positive integers. A **matrix**, A, is a rectangular array having m rows and n columns of elements in a field F. In particular, it is represented as follows:

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \tag{2.11}$$

If the field $F = \mathbb{R}$, then $\mathrm{A} \in \mathbb{R}^{m \times n}$. If $m = n$ the matrix is called **square**. The set of entries A_{ij} with $i = j$ is called **main diagonal** of the matrix A.
Moreover:

- Let $\mathrm{A} \in F^{m \times n}$. The maximum number of linearly independent columns (or rows) of A is called **rank** (rank(A)). A is said to have a **complete rank** if $\mathrm{rank(A)} = \min(m, n)$.
- Considering $\mathrm{A} \in \mathbb{R}^{m \times p}$, $\mathrm{B} \in \mathbb{R}^{m \times p}$, $\mathrm{C} \in \mathbb{R}^{p \times n}$, $\lambda \in F$, one can define the following operations between such matrices:
  - Matrix addition:

$$\mathrm{A} + \mathrm{B} = (a_{ij} + b_{ij}), \mathrm{A} + \mathrm{B} \in \mathbb{R}^{m \times p} \tag{2.12}$$

  - Matrix multiplication by a scalar:

$$\lambda \mathrm{A} = (\lambda a_{ij}), \lambda \mathrm{A} \in \mathbb{R}^{m \times p} \tag{2.13}$$

  - Matrix multiplication:

$$\mathrm{AC} = \left(\sum_{k=1}^{p} a_{ik} b_{kj} \right), \mathrm{A} \cdot \mathrm{C} \in \mathbb{R}^{m \times n} \tag{2.14}$$

    In particular, matrix multiplication is defined only when the number of columns of the first matrix is equal to the number of rows of the second matrix. Matrix product is associative and distributive with respect to matrix addition, but it is not, in general, commutative $(\mathrm{A} \cdot \mathrm{C} \neq \mathrm{C} \cdot \mathrm{A})$.
  - Transposition:

$$\mathrm{A}^T = (a_{ji}), \mathrm{A}^T \in \mathbb{R}^{p \times m} \tag{2.15}$$

- A **lower triangular matrix** is a matrix L where $l_{ij} = 0$ if $i < j$. On the other hand, an **upper triangular matrix** is a matrix U where $u_{ij} = 0$ if $i > j$.
- A **diagonal matrix** of order n is a square matrix A where $a_{ij} \neq 0$ if $i = j$ and 0 otherwise.
- An **identity matrix** of order n is a square matrix I_n where $a_{ij} = 1$ if $i = j$ and 0 otherwise. The identity matrix is the only matrix such that $\mathrm{AI}_n = \mathrm{I}_n\mathrm{A}$ for all possible A.
- A matrix is called **invertible** or **nonsingular** if there exists a matrix $\mathrm{B}(n \times n)$ such that $\mathrm{AB} = \mathrm{I}$. In particular, the matrix B is the **inverse** matrix of A and is denoted as A^{-1}. Moreover, if A is invertible, then $(\mathrm{A}^T)^{-1} = (\mathrm{A}^{-1})^T = \mathrm{A}^{-T}$.
  - A square matrix is invertible if and only if its column vectors are linearly independent.

- A square matrix A is **symmetric** if $\mathrm{A} = \mathrm{A}^T$, while it is **anti-symmetric** if $\mathrm{A} = -\mathrm{A}^T$. Moreover, it is **symmetric, positive, definite** if, given a vector space V:

$$\forall \boldsymbol{x} \in V \setminus \{\mathbf{0}\} : \boldsymbol{x}^T \mathrm{A} \boldsymbol{x} > 0 \tag{2.16}$$

  One can obtain a symmetric, positive, definite matrix S from a matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ by performing $\mathrm{S} = \mathrm{A}^T\mathrm{A}$ when $\mathrm{rank}(\mathrm{A}) = n$.

- Let $\mathrm{A} \in \mathbb{R}^{n \times n}$. The **determinant** of A ($\det(A)$) is the scalar defined as:

$$\det(\mathrm{A}) = \begin{cases} a_{11} & \text{if } n = 1 \\ \sum_{j=1}^{n} (-1)^{i+j} \det(\mathrm{A}_{ij}) a_{ij} = \sum_{i=1}^{n} (-1)^{i+j} \det(\mathrm{A}_{ij}) a_{ij} & \text{if } n > 1 \end{cases} \tag{2.17}$$

  with $i \in \{1, \dots n\}$ ($j \in 1, \dots, n$) fixed, where A_{ij} if the sub-matrix of order $n-1$ obtained from A by eliminating row i and column j.

  - If A is diagonal or triangular, then:

$$\det(\mathrm{A}) = \prod_{i=1}^{n} a_{ii} \tag{2.18}$$

  - The determinant has the following properties:

$$\det(\mathrm{A}) = \det(\mathrm{A}^T) \tag{2.19}$$
$$\det(\mathrm{AB}) = \det(\mathrm{A}) \det(\mathrm{B}) \tag{2.20}$$
$$\det(\mathrm{A}^{-1}) = \frac{1}{\det(\mathrm{A})} \tag{2.21}$$
$$\det(\alpha \mathrm{A}) = \alpha^n \det(\mathrm{A}), \forall \alpha \in F \tag{2.22}$$

  - Every orthogonal matrix A is invertible and $\det(\mathrm{A}) = 1$.

- For a matrix A the following properties are equivalent:

  - A is nonsingular.
  - $\det(\mathrm{A}) \neq 0$.
  - $\mathrm{rank}(\mathrm{A}) = n$.
  - A has linearly independent columns and rows.

- Let $\mathrm{A} \in \mathbb{R}^{n \times n}$. The number $\lambda \in \mathbb{R}$ is called an **eigenvalue** of A if:

$$\exists \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \mathbf{0} \text{ such that } \mathrm{A}\boldsymbol{x} = \lambda \boldsymbol{x} \tag{2.23}$$

  The vector $\boldsymbol{x}$ is the **eigenvector** associated with the eigenvalue λ. The set of eigenvalues of A is called **spectrum** of A ($\sigma(\mathrm{A})$).

- The maximum eigenvalue in module of A is called the **spectral radius** of A ($\rho(\mathrm{A}) = \max_{\lambda \in \sigma(\mathrm{A})} |\lambda|$).
- If A has n eigenvalues:

$$\det(\mathrm{A}) = \prod_{i=1}^{n} \lambda_i \tag{2.24}$$

- A matrix is singular if and only if it has at least one null eigenvalue.
- If A has n eigenvalues and it is either diagonal or triangular, then $\lambda_i = a_{ii}, i = 1, \ldots, n$.
- A symmetric positive definite (respectively semi-definite) matrix A has eigenvalues λ_i greater (respectively equal) than zero.
- Two matrices, $\mathrm{A}, \mathrm{B} \in \mathbb{R}^{n \times n}$, are said to be **similar** if there exists a nonsingular matrix P such that $\mathrm{B} = \mathrm{PAP}^{-1}$. In other terms, the matrices A, B are similar if and only if they have the same eigenvalues.

2.3 Scalar Product and Norms in Vector Spaces

- The scalar product, or inner product, on a vector space V is a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}^+$ which has the following properties:
  - $\langle \alpha \boldsymbol{x} + \beta \boldsymbol{y}, \boldsymbol{z} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \beta \langle \boldsymbol{y}, \boldsymbol{z} \rangle$, $\forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$, $\forall \alpha, \beta \in \mathbb{R}$.
  - $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$, $\forall \boldsymbol{x}, \boldsymbol{y} \in V$.
  - $\langle \boldsymbol{x}, \boldsymbol{x} \rangle > \boldsymbol{0}$, $\forall \boldsymbol{x} \neq \boldsymbol{0}$.

  In particular, if $V = \mathbb{R}^n$, then:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i \tag{2.25}$$

- Let V be a vector space over the field F. The function $\| \cdot \| : V \to F$ is the **norm** of V if the following properties are satisfied:
  - $\|\boldsymbol{v}\| \geq 0, \forall \boldsymbol{v} \in V$ and $\|\boldsymbol{v}\| = 0 \Rightarrow \boldsymbol{v} = \boldsymbol{0}$.
  - $\|\alpha \boldsymbol{v}\| = |\alpha| \|\boldsymbol{v}\|, \forall \alpha \in F, \forall \boldsymbol{v} \in V$.
  - $\|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\|, \forall \boldsymbol{v}, \boldsymbol{w} \in V$.

  In particular:
  - The **one norm** of a vector $\boldsymbol{v} \in V$ of n elements is computed as:

$$\|\boldsymbol{v}\|_1 = \sum_{i=1}^{n} |v_i| \tag{2.26}$$

– The p-**norm** of a vector $\boldsymbol{v} \in V$ of n elements is computed as:

$$\|\boldsymbol{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}, 1 \leq p < \infty \tag{2.27}$$

The 2-norm of a vector $\boldsymbol{x}$ is equivalent to $\langle \boldsymbol{x}, \boldsymbol{x} \rangle$.

– The **infinity norm** of a vector $\boldsymbol{v} \in V$ of n elements is computed as:

$$\|\boldsymbol{v}\|_\infty = \max_{i=1,\ldots,n} |v_i| \tag{2.28}$$

– Two norms $\|\cdot\|_p$ and $\|\cdot\|_q$ are said to be equivalent if there exists two positive constants c_{pq} and C_{pq} such that:

$$c_{pq}\|\boldsymbol{x}\|_q \leq \|\boldsymbol{x}\|_p \leq C_{pq}\|\boldsymbol{x}\|_q, \forall \boldsymbol{x} \in V \tag{2.29}$$

∗ In a vector space, all the p-norms are equivalent.

- A matrix norm is a function $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ satisfying the following properties:

  – $\|\mathrm{A}\| \geq 0, \forall \mathrm{A} \in \mathbb{R}^{m \times n}$ and $\|\mathrm{A}\| = 0 \Leftrightarrow \mathrm{A} = \mathbf{0}$.
  – $\|\alpha \mathrm{A}\| = |\alpha| \|\mathrm{A}\|, \forall \alpha \in F, \forall \mathrm{A} \in \mathbb{R}^{m \times n}$.
  – $\|\mathrm{A} + \mathrm{B}\| \leq \|\mathrm{A}\| + \|\mathrm{B}\|, \forall \mathrm{A}, \mathrm{B} \in \mathbb{R}^{m \times n}$.

  Considering $\mathrm{A} \in \mathbb{R}^{m \times n}$:

  – The **spectral norm** of A is:

$$\|\mathrm{A}\|_2 = \sqrt{\rho(\mathrm{A}^T \mathrm{A})} \tag{2.30}$$

  – The p-**norm** of A, with $p = 1$, is:

$$\|\mathrm{A}\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}| \tag{2.31}$$

  – The **infinity norm** of A is:

$$\|\mathrm{A}\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}| \tag{2.32}$$

  – The **Frobenius norm** of A is:

$$\|\mathrm{A}\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2} \tag{2.33}$$

2.4 Linear Mappings

Linear mappings are mappings on vector spaces that preserve their structure, which allow one to define the concept of coordinate. In particular, considering two real vector spaces V and W, a mapping $\Phi : V \to W$ preserves the structure of the vector space if:

$$\Phi(\boldsymbol{x} + \boldsymbol{y}) = \Phi(\boldsymbol{x}) + \Phi(\boldsymbol{y}) \tag{2.34}$$

$$\Phi(\lambda \boldsymbol{x}) = \lambda \Phi(\boldsymbol{x}) \tag{2.35}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in V$ and $\lambda \in \mathbb{R}$. Equivalently:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in V \; \forall \lambda, \psi \in \mathbb{R} : \Phi(\lambda \boldsymbol{x} + \psi \boldsymbol{y}) = \lambda \Phi(\boldsymbol{x}) + \psi \Phi(\boldsymbol{y}) \tag{2.36}$$

Linear mappings are representable as matrices. Considering a vector space V and an ordered basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of V, then for any $\boldsymbol{x} \in V$ one can obtain a unique representation (i.e. linear combination)

$$\boldsymbol{x} = \alpha_1 \boldsymbol{b}_1 + \cdots + \alpha_n \boldsymbol{b}_n \tag{2.37}$$

of $\boldsymbol{x}$ with respect to B. Then, $\alpha_1, \ldots, \alpha_n$ are the **coordinates** of $\boldsymbol{x}$ with respect to B, and the vector

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \in \mathbb{R}^n \tag{2.38}$$

is the **coordinate vector / coordinate representation** of $\boldsymbol{x}$ with respect to the ordered basis B. In other words, a basis effectively defines a coordinate system. For example, considering a geometric vector $\boldsymbol{x} \in \mathbb{R}^2$ with coordinates $[2, 3]^T$ with respect to the standard basis $(\boldsymbol{e}_1, \boldsymbol{e}_2)$ (i.e. $\boldsymbol{e}_1 = [1, 0]^T$, and $\boldsymbol{e}_2 = [0, 1]^T$) of $\mathbb{R}^2$, one can write:

$$\boldsymbol{x} = 2\boldsymbol{e}_1 + 3\boldsymbol{e}_2$$

However, if one considers the basis $(\boldsymbol{b}_1, \boldsymbol{b}_2)$, with $\boldsymbol{b}_1 = [1, -1]^T$ and $\boldsymbol{b}_2 = [1, 1]^T$, one would obtain the coordinates $\frac{1}{2}[-1, 5]^T$ to represent the same vector with respect to this new basis.

2.4.1 Transformation Matrix

Considering the vector spaces V and W with corresponding bases $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m)$ and a linear mapping $\Phi : V \to W$, for $j \in \{1, \ldots, n\}$ one has that

$$\Phi(\boldsymbol{b}_j) = \alpha_{ij} \boldsymbol{c}_1 + \cdots + \alpha_{mj} \boldsymbol{c}_m = \sum_{i=1}^{m} \alpha_{ij} \boldsymbol{c}_i \tag{2.39}$$

is the unique representation of $\Phi(\boldsymbol{b}_j)$ with respect to C. In particular, the $m \times n$ matrix A_Φ, whose elements are given by

$$\mathrm{A}_\Phi(i,j) = \alpha_{ij} \tag{2.40}$$

is called the **transformation matrix** of Φ. If, for example, $\hat{\boldsymbol{x}}$ is the coordinate vector of $\boldsymbol{x} \in V$ with respect to B and $\hat{\boldsymbol{y}}$ the coordinate vector of $\boldsymbol{y} = \Phi(x) \in W$ with respect to W, then:

$$\hat{\boldsymbol{y}} = \mathrm{A}_\Phi \hat{\boldsymbol{x}} \tag{2.41}$$

This means that the transformation matrix can be used to map coordinates with respect to an ordered basis in V to coordinates with respect to an ordered basis in W.

2.5 Affine Spaces

Affine spaces are spaces that are offset from the origin, i.e. spaces that are no longer vector subspaces. In particular, considering a vector space V, $\boldsymbol{x}_0 \in V$ and a subspace $U \subseteq V$, then the subset

$$L = \boldsymbol{x}_0 + U \coloneqq \{\boldsymbol{x}_0 + \boldsymbol{u} : \boldsymbol{u} \in U\} \tag{2.42}$$

$$= \{\boldsymbol{v} \in V | \exists \boldsymbol{u} \in U : \boldsymbol{v} = \boldsymbol{x}_0 + \boldsymbol{u}\} \subseteq V \tag{2.43}$$

is called **affine subspace** or **linear manifold** of V, while U is called **direction** or **direction space**, and $\boldsymbol{x}_0$ is called **support point**. Thus, an affine subspace excludes $\mathbf{0}$ if $\boldsymbol{x}_0 \notin U$, hence is not a linear vector subspace of V. Sometimes these spaces are called **hyperplanes**. Examples of affine subspaces are points, lines and planes in $\mathbb{R}^3$, since they do not necessarily pass through the origin.

2.6 Affine Mappings

For two vector spaces V and W, a linear mapping $\Phi : V \to W$, and $\boldsymbol{a} \in W$, the mapping

$$\phi : V \to W \tag{2.44}$$

$$x \mapsto \boldsymbol{a} + \Phi(\boldsymbol{x}) \tag{2.45}$$

is an **affine mapping** from V to W. In particular, the vector $\boldsymbol{a}$ is called the **translation vector** of ϕ.

2.7 Angles and Orthogonality

In addition to enabling the definition of lengths of vectors, as well as the distance between two vectors, the inner product also captures the geometry of a vector space by defining the angle ω between two vectors. In particular, given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\cos\omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} \tag{2.46}$$

In particular:

- Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are **orthogonal** if and only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, i.e. if they are perpendicular ($\boldsymbol{x} \perp \boldsymbol{y}$). In addition, if $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$, i.e. the vectors are **unit vectors**, then they are said to be **orthonormal** vectors.

- A square matrix $\mathrm{A} \in \mathbb{R}^{n \times n}$ is an **orthogonal matrix** if and only if its columns are orthonormal so that:

$$\mathrm{A}\mathrm{A}^{-1} = \mathrm{I} = \mathrm{A}^T\mathrm{A} \quad \Rightarrow \quad \mathrm{A}^{-1} = \mathrm{A}^T \tag{2.47}$$

  This can be checked by verifying that by taking two column vectors of the given matrix:

$$\boldsymbol{a}_i^T \boldsymbol{a}_j = \langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle = \begin{cases} 0 & \text{if } i \neq j, \text{ i.e. the two vectors are orthogonal} \\ 1 & \text{if } i = j, \text{ i.e. the two vectors are unit vectors} \end{cases} \tag{2.48}$$

- Considering an n-dimensional vector space V and a basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ of V is an **orthonormal basis**, if the vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ are orthonormal, i.e.:

$$\boldsymbol{b}_i^T \boldsymbol{b}_j = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = \begin{cases} 0 & \text{if } i \neq j, \text{ i.e. the two vectors are orthogonal} \\ 1 & \text{if } i = j, \text{ i.e. the two vectors are unit vectors} \end{cases} \tag{2.49}$$

2.7.1 Orthogonal Projections

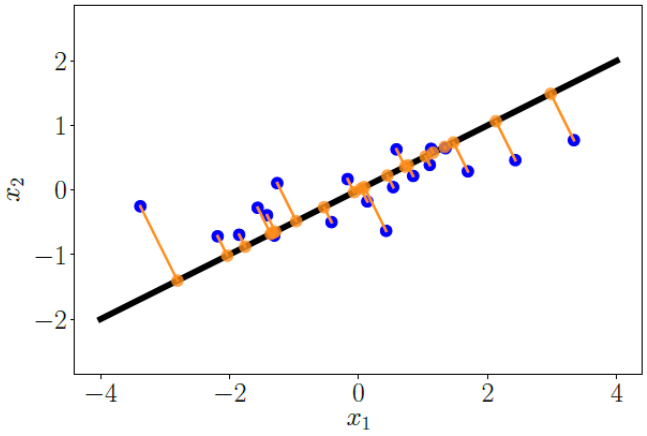
Orthogonal projections are linear transformations (e.g. rotations). High-dimensional data is often hard to analyse or visualize. However, high-dimensional data quite often possesses the property that only a few dimensions contain most information, and most other dimensions are not essential to describe key properties of the data. More specifically, one can project the original high-dimensional data onto a lower-dimensional feature space and work in this lower-dimensional space to learn more about the dataset and extract relevant patterns. For example, the figure below shows an orthogonal projection (orange dots) of a two-dimensional dataset (blue dots) onto a one-dimensional subspace (straight line). It is also possible to notice that, the distance of the points to the straight line correspond to the minimal possible distances, hence the projections of all the points are orthogonal to the one-dimensional subspace (otherwise, if the projection were not orthogonal to the straight line, the distances would not be minimal).

- Let V be a vector space and $U \subseteq V$ a subspace of V. A linear mapping $\pi : V \to U$ is called **projection** if:

$$\pi^2 = \pi \circ \pi = \pi \tag{2.50}$$

  In particular, a projection can be seen as a compression. There is always a loss of information.

- Since linear mappings can be expressed by transformation matrices, projections can also be expressed by **projections matrices**, P, which exhibit the property $\mathrm{P}^2 = \mathrm{P}$.



- Let $\boldsymbol{x} \in \mathbb{R}^n$, $U \subseteq \mathbb{R}^n$, with $\dim(U) = m \geq 1$ and $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ an ordered basis of U. Any projection $\pi_U(\boldsymbol{x})$ onto U is necessarily an element of U (and of $\mathbb{R}^n$ since U is a subspace of it), therefore, it can be represented as linear combination of the basis vectors, such that:

$$\pi_U(\boldsymbol{x}) = \sum_{i=1}^{m} \lambda_i \boldsymbol{b_i}, \ \lambda \in \mathbb{R} \tag{2.51}$$

  It can be shown that the projection of $\boldsymbol{x}$ onto U is computed as follows:

$$\pi_U(\boldsymbol{x}) = \mathrm{B}(\mathrm{B}^T\mathrm{B})^{-1}\mathrm{B}^T\boldsymbol{x}, \quad \mathrm{P}_\pi = \mathrm{B}(\mathrm{B}^T\mathrm{B})^{-1}\mathrm{B}^T \tag{2.52}$$

- The **projection error** is the distance between $\boldsymbol{x}$ and its projection (i.e. $\boldsymbol{x} - \pi(\boldsymbol{x})$).

- A projection can be seen from a different point of view. Projections allow one to look at situations where one has a linear system $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$ with no solutions. In particular A has more rows than columns (i.e. one has an overdetermined system, with more equations than unknowns). In this case, one can find a solution $\tilde{\boldsymbol{x}}$ which approximate the solution $\boldsymbol{x}$. Thus:

$$\mathrm{A}\tilde{\boldsymbol{x}} \approx \boldsymbol{b} \quad \Rightarrow \quad \mathrm{A}\tilde{\boldsymbol{x}} - \boldsymbol{b} \neq \boldsymbol{0} \tag{2.53}$$

  The quantity $\boldsymbol{r} = \mathrm{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}$ is called **residual**. At this point, one can measure $\boldsymbol{r}$ using a 2-norm and minimize this norm (i.e. $\min_{\boldsymbol{x}\in\mathbb{R}^m} \|\mathrm{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_2^2$). This solution is called **least-square solution**.

2.8 Matrix Decompositions

Matrix factorizations, or decompositions, write a given matrix A as the product of matrices. In particular, special matrices used in matrix decompositions are triangular and diagonal matrices.

2.8.1 Gaussian Elimination Method or LU Factorization

Let $\mathrm{A} \in \mathbb{R}^{n \times n}$. If A is a nonsingular and all the principal minors A_i, $i = 1, \ldots, n$, are nonsingular, then there exist two matrices L and U, with L lower triangular with $l_{ii} = 1, i = 1, \ldots, n$, and U upper triangular nonsingular, such that:

$$\mathrm{A} = \mathrm{LU} \tag{2.54}$$

This is called LU-**factorization** of A. In particular, the matrices L and U can be computed in $n - 1$ steps with the Gaussian Elimination Method (GEM) having computational cost of $O(n^3/3)$. One problem of the GEM algorithm is that it is unstable, i.e. the algorithmic error is not limited. To avoid this, the **pivoting technique** is employed. Applying this technique is the same as applying a left multiplication by a permutation matrix P. Hence:

$$\mathrm{PA} = \mathrm{LU} \tag{2.55}$$

Moreover, if A is a symmetric, positive and semi-definite matrix, then:

$$\mathrm{A} = \mathrm{LL}^T \tag{2.56}$$

2.8.2 Eigendecomposition

Eigendecomposition is a factorization which involves a diagonal matrix instead of triangular matrices. In particular, let $\mathrm{A} \in \mathbb{R}^{m \times n}$. Hence, A can be decomposed as:

$$\mathrm{A} = \mathrm{PDP}^{-1} \tag{2.57}$$

where P is a nonsingular matrix of size $n \times n$ and D is a diagonal matrix with diagonal elements d_{ii} corresponding to the eigenvalues of A if and only if the eigenvectors of A are linear independent and form a basis of $\mathbb{R}^n$. This decomposition can be applied only to square matrices and with particular properties on their spectrum.

2.8.3 Singular Value Decomposition (SVD)

Also this factorization involves a diagonal matrix and can be applied to all matrices. In particular, any matrix $A \in \mathbb{R}^{m \times n}, m \geq n$ (works also with every other condition on m and n), with $\mathrm{rank(A)} = k, k \leq n$ can be written as:

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^T \tag{2.58}$$

where:

- $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix with orthogonal vectors $\boldsymbol{u}_i$.
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix with orthogonal vectors $\boldsymbol{v}_i$.

- $\Sigma \in \mathbb{R}^{m \times n}$ is a matrix whose diagonal entries are the **singular values** σ_i of A and with extra-diagonal entries equal to zero. In particular:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \sigma_{k+1} = \sigma_{k+2} = \cdots = \sigma_n = 0 \tag{2.59}$$

  where $k = \text{rank}(\text{A})$. The singular matrix Σ is unique, while the other two matrices are not.

This decomposition performs two changes of basis, via the orthogonal matrices U and V (which can be seen as linear mappings) and a scaling operation via the matrix Σ. Moreover, it can be shown that, since:

$$\begin{aligned} \text{A}^T\text{A} &= (\text{U}\Sigma\text{V}^T)^T(\text{U}\Sigma\text{V}^T) = \text{V}\Sigma^T\text{U}^T\text{U}\Sigma\text{V}^T \\ &= \Sigma^T\Sigma \\ &= \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix} \end{aligned} \tag{2.60}$$

then:

$$\sigma_i = \sqrt{\lambda_i(\text{A}^T\text{A})},\ i = 1, \ldots, n \tag{2.61}$$

where $\lambda_i(\text{A}^T\text{A})$ is the i-th eigenvalue of the matrix A^TA. In particular, the maximum singular value σ_1 corresponds to:

$$\sigma_1 = \sqrt{\lambda_{max}(\text{A}^T\text{A})} = \sqrt{\rho(\text{A}^T\text{A})} = \|\text{A}\|_2 \tag{2.62}$$

while the minimum singular value σ_n corresponds to:

$$\sigma_n = \sqrt{\lambda_{min}(\text{A}^T\text{A})} \tag{2.63}$$

from which:

$$\|\text{A}^{-1}\| = \frac{1}{\sigma_n} \tag{2.64}$$

Thus:

$$K(\text{A}) = \frac{\sigma_1}{\sigma_n} \tag{2.65}$$

2.8.4 Matrix Approximation

Given the SVD of a matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$, $\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^T$, one can use it to approximate the matrix A as a sum of low-rank matrices $\mathrm{A}_i \in \mathbb{R}^{m \times n}$ with $\mathrm{rank}(\mathrm{A}_i) = 1$ such that:

$$\mathrm{A}_i = \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{2.66}$$

The matrix of $\mathrm{rank}(\mathrm{A}) = k$ can then be written as:

$$\mathrm{A} = \sum_{i=1}^{k} \sigma_i \mathrm{A}_i = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{2.67}$$

To obtain a rank-p approximation, $\mathrm{A}_p, p < k$ of A, one can truncate the sum at index $i = p$. The error introduced with this approximation can be computed as:

$$\|\mathrm{A} - \mathrm{A}_p\|_2 = \left\| \sum_{i=p+1}^{k} \sigma_i \mathrm{A}_i \right\|_2 = \sigma_{p+1} \tag{2.68}$$

2.9 Linear Systems

A linear system can be written as:

$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.69}$$

where A is a matrix of size $m \times n$ (with $m \geq n$), $\boldsymbol{x}$ is a column vector of length n and $\boldsymbol{b}$ is a column vector of length m. In particular, if $m = n$, the system of equations is called **square linear system**. The solution of a square linear system exists and is unique if and only if one of the following conditions is met:

- A is nonsingular ($\det(\mathrm{A}) \neq 0$).
- $\mathrm{rank}(\mathrm{A}) = n$.
- The system $\mathrm{A}\boldsymbol{x} = \mathbf{0}$ admits only the solution $\boldsymbol{x} = \mathbf{0}$.

Such solution can be computed as:

$$\boldsymbol{x} = \mathrm{A}^{-1}\boldsymbol{b} \tag{2.70}$$

Another method to compute the solution of such a system is by applying the **Cramer's rule**:

$$x_i = \frac{\det(\mathrm{A}_i)}{\det(\mathrm{A})}, i = 1, \ldots, n \tag{2.71}$$

where A_i is obtained from A by replacing the i-th column by $\boldsymbol{b}$. However, both of these methods are computationally expensive. Thus, in order to find a solution, one can rely on two types of methods:

- **Direct methods**, which yield a more precise solution in a finite number of steps.
- **Iterative methods**, which yield a less precise solution and require an infinite number of steps.

Moreover, it is important to perform an error analysis on the obtained results:

- **Rounding** or **arithmetic errors**. These depend on the algorithm steps. An algorithm which produces an arithmetic error limited by a constant is called **stable**.
- **Inherent errors**. These are due to errors in the data representation and do not depend on the algorithm. If the inherent error is large, one has a **ill-posed problem**. The analysis of inherent errors is performed by supposing to use exact arithmetic to compute the result. For example, if A or $\boldsymbol{b}$ slightly change (e.g. due to machine precision) the solution $\boldsymbol{x}$ also changes. In particular, one is interested in comparing $\left\|\frac{\Delta \boldsymbol{x}}{\boldsymbol{x}}\right\|$ with $\left\|\frac{\Delta \mathrm{A}}{\mathrm{A}}\right\|$ and $\left\|\frac{\Delta \boldsymbol{b}}{\boldsymbol{b}}\right\|$ to see how much the solution has changed. It can be demonstrated that:

$$\left\|\frac{\Delta \boldsymbol{x}}{\boldsymbol{x}}\right\| \leq K(\mathrm{A})\left(\left\|\frac{\Delta \mathrm{A}}{\mathrm{A}}\right\| + \left\|\frac{\Delta \boldsymbol{b}}{\boldsymbol{b}}\right\|\right), \quad K(\mathrm{A}) = \|\mathrm{A}\|\|\mathrm{A}^{-1}\| \tag{2.72}$$

  In particular, $K(\mathrm{A})$ is the **condition number** of the matrix A. The condition number measures how sensitive a function is to changes/errors in the input and how much the output changes as a result of the operations performed. A **well-conditioned** system is a system where $K(\mathrm{A})$ is small, while an **ill-conditioned** system is a system where $K(\mathrm{A})$ is large.

2.9.1 Direct Methods

Let $\mathrm{A} = \mathrm{LU}$. The solution of the linear system $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$ can be computed by solving two triangular systems:

$$\mathrm{L}\boldsymbol{y} = \boldsymbol{b} \quad \text{forward substitutions algorithm} \tag{2.73}$$
$$\mathrm{U}\boldsymbol{x} = \boldsymbol{y} \quad \text{backward substitutions algorithm} \tag{2.74}$$

The advantage is that these two systems have triangular matrices. The solution to such methods have $O(n^2/2)$ complexity. In the case of pivoting technique:

$$\mathrm{L}\boldsymbol{y} = \mathrm{P}\boldsymbol{b} \quad \text{forward substitutions algorithm} \tag{2.75}$$
$$\mathrm{U}\boldsymbol{x} = \boldsymbol{y} \quad \text{backward substitutions algorithm} \tag{2.76}$$

2.9.2 Iterative Methods

Iterative methods are numerical methods that approximate a solution using a procedure involving a number of steps which tends to infinity. The basic idea is to construct a sequence of vectors $\boldsymbol{x}_k$ which has the property of convergence:

$$\boldsymbol{x}^* = \lim_{k \to \infty} \boldsymbol{x}_k \tag{2.77}$$

where $\boldsymbol{x}^*$ is the exact solution and the starting guess $\boldsymbol{x}_0$ is given. In general, the sequence $\boldsymbol{x}_k$ is obtained as $\boldsymbol{x}_k = g(\boldsymbol{x}_{k-1}), k = 1, \ldots$ where g is a particular function or set of operations acting on $\boldsymbol{x}_{k-1}$. For example, the gradient descent method can be applied to solve such systems.

Chapter 3

Numerical Optimization

Machine learning models are expressed as minimization problems. In order to train such models one need to compute the best parameters for the training data set. The definition of *best* is modelled by the optimization of a given objective function, usually called **loss function**.

3.1 Multivariate Functions

- Let $f : \mathbb{R}^n \to \mathbb{R}$, then f is **differentiable** with respect to x_i, if the limit

$$\lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n)}{h} = \frac{\partial f}{\partial x_i}(\boldsymbol{x}) \tag{3.1}$$

  exists.

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. Then, the vector

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}(\boldsymbol{x}), \ldots, \frac{\partial f}{\partial x_n}(\boldsymbol{x}) \right) \tag{3.2}$$

  is called **gradient** of f.

- The basic rules of partial differentiation are the following:

$$\text{Product rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}(f(\boldsymbol{x})g(\boldsymbol{x})) = \frac{\partial f}{\partial \boldsymbol{x}}g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}} \tag{3.3}$$

$$\text{Sum rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}(f(\boldsymbol{x}) + g(\boldsymbol{x})) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}} \tag{3.4}$$

$$\text{Chain rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}(g(f(\boldsymbol{x}))) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}} \tag{3.5}$$

  - Considering $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x_1 and x_2. Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t. To compute the gradient of f with respect to t, one need to apply the chain rule for multivariate functions as:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{3.6}$$

where d denotes the gradient and ∂ partial derivatives. For example, considering $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$, then:

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}t} &= \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \\ &= 2\sin t\frac{\partial \sin t}{\partial t} + 2\frac{\partial \cos t}{\partial t} \\ &= 2\sin t\cos t - 2\sin t \\ &= 2\sin t(\cos t - 1) \end{aligned}$$

is the corresponding derivative of f with respect to t. In this case, the gradient is in $\mathbb{R}$.

– Considering a function $f(x_1, x_2)$ where $x_1(s, t)$ and $x_2(s, t)$ are themselves functions of two variables, s and t, the chain rule yields the partial derivatives:

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s} \tag{3.7}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{3.8}$$

$$\tag{3.9}$$

and the gradient is obtained by the matrix multiplication:

$$\frac{\mathrm{d}f}{\mathrm{d}(s,t)} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial (s,t)} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix} \tag{3.10}$$

– Considering a function $f : \mathbb{R}^n \to \mathbb{R}^m$, where $n \geq 1$ and $m > 1$, and a vector $\boldsymbol{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$, the corresponding vector of function values is given as:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^m \tag{3.11}$$

$\boldsymbol{f}$ is called a **vector-valued function**. Each element of $\boldsymbol{f}$ is a function $f_i : \mathbb{R}^n \to \mathbb{R}$. The gradient of $\boldsymbol{f}$ with respect to $\boldsymbol{x} \in \mathbb{R}^n$ is:

$$\boldsymbol{J} = \nabla_{\boldsymbol{x}}\boldsymbol{f} = \frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{3.12}$$

In particular, the collection of all first-order derivatives of a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is called the **Jacobian**.

- In many machines learning applications, one finds good model parameters by performing gradient descent, which relies on the fact that one can compute the gradient of a learning objective with respect to the parameters of the model. However, writing out the gradient is often impractical. In particular, considering a multilayer neural network, the output value in computed as a multi-level function composition of the following form:

$$\boldsymbol{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\boldsymbol{x}) \tag{3.13}$$

where each layer can be seen as a function that takes in input the previous layer's output:

$$f_i(\boldsymbol{x}_{i-1}) = \sigma(\boldsymbol{A}_{i-1}\boldsymbol{x}_{i-1} + \boldsymbol{b}_{i-1}) \tag{3.14}$$

In order to train these models, one requires the gradient of a loss function L with respect to all model parameters $\boldsymbol{A}_j$, $\boldsymbol{b}_j$ for $j = 1, \ldots, K$. This also requires one to compute the gradient of L with respect to the inputs of each layer. Thus, considering such networks, one may be interested in finding $\boldsymbol{A}_j$, $\boldsymbol{b}_j$ such that the squared loss:

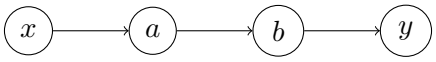
$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\theta, \boldsymbol{x})\|^2 \tag{3.15}$$

is minimized, where $\theta = \{\boldsymbol{A}_0, \boldsymbol{b}_0, \ldots, \boldsymbol{A}_{K-1}, \boldsymbol{b}_{K-1}\}$. To obtain the gradients with respect to such parameters, the chain rule can be applied:

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial \boldsymbol{f}_K}\frac{\partial L \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}}\frac{\partial \boldsymbol{f}_{i+1}}{\partial \theta_i} \tag{3.16}$$

where $\theta_i = \{\boldsymbol{A}_i, \boldsymbol{b}_i\}$. This process is called **backpropagation**.

- It turns out that backpropagation is a special case of **automatic differentiation**. This technique can be seen as a process to numerically evaluate the exact gradient of a function by working with intermediate variables and applying the chain rule. Automatic differentiation has forward and reverse modes, depending on how gradients are propagated during the application of the chain rule. For example, considering the following graph:



if one were to compute the derivative $\frac{\mathrm{d}y}{\mathrm{d}x}$, one could apply the chain rule in two different ways:

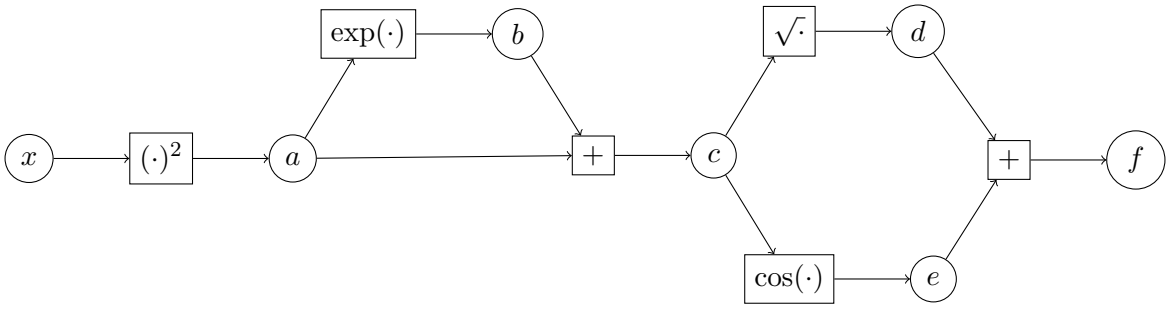
$$\frac{\mathrm{d}y}{\mathrm{d}x} = \left(\frac{\mathrm{d}y}{\mathrm{d}b}\frac{\mathrm{d}b}{\mathrm{d}a}\right)\frac{\mathrm{d}a}{\mathrm{d}x} \tag{3.17}$$

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}b}\left(\frac{\mathrm{d}b}{\mathrm{d}a}\frac{\mathrm{d}a}{\mathrm{d}x}\right) \tag{3.18}$$

The former way corresponds to the forward automatic differentiation technique, while the latter to the backward automatic differentiation technique (i.e. backpropagation). For example, considering the function $f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2))$, one can define the following intermediate variables:

$$\begin{aligned} a &= x^2 \\ b &= \exp(a) \\ c &= a + b \\ d &= \sqrt{c} \\ e &= \cos(c) \\ f &= d + e \end{aligned}$$

and represent such function using the following graph:



so that:

$$
\begin{aligned}
\frac{\partial a}{\partial x} &= 2x \\
\frac{\partial b}{\partial a} &= \exp(a) \\
\frac{\partial c}{\partial a} &= 1 = \frac{\partial c}{\partial b} \\
\frac{\partial d}{\partial c} &= \frac{1}{2\sqrt{c}} \\
\frac{\partial e}{\partial c} &= -\sin(c) \\
\frac{\partial f}{\partial d} &= 1 = \frac{\partial f}{\partial e}
\end{aligned}
$$

By looking at the computation graph, one can compute $\partial f/\partial x$ by working backward and obtain:

$$
\begin{aligned}
\frac{\partial f}{\partial c} &= \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c} \\
\frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c}\frac{\partial c}{\partial b} \\
\frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a} \\
\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}
\end{aligned}
$$

At this point, it is possible to easily compute $\partial f/\partial x$. Automatic differentiation is a formalization of such example. Let $x_1, \ldots, x_d$ be the input variables to the given function, $x_{d+1}, \ldots, x_{D-1}$ be the intermediate variables, and x_D the output variable. Then, the computation graph can be expressed as:

$$
\text{For } i = d+1, \ldots, D: \quad x_i = g_i(x_{\text{Pa}(x_i)}) \tag{3.19}
$$

where the $g_i(\cdot)$ are elementary functions and $x_{\text{Pa}(x_i)}$ are the parent nodes of the variable x_i in the graph. Given such a function, one can use the chain rule to compute the derivative of the function as follows:

$$
\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in \text{Pa}(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in \text{Pa}(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial g_j}{\partial x_i} \tag{3.20}
$$

where $\text{Pa}(x_j)$ is the set of parent nodes of x_j in the graph. In particular, equation 3.20 corresponds to the backpropagation algorithm, while the 3.19 equation corresponds to the forward automatic differentiation technique.

- Sometimes, one might be interested in derivatives of higher order. Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a function of two variables x and y. The **higher-order derivatives** of such function are the following:
  - $\frac{\partial^2 f}{\partial x^2}$ is the second partial derivative of f with respect to x.
  - $\frac{\partial^n f}{\partial x^n}$ is the n-th partial derivative of f with respect to x.
  - $\frac{\partial^2 f}{\partial y \partial x}$ is the second partial derivative of f obtained by first partial differentiating with respect to x and then with respect to y.
  - $\frac{\partial^2 f}{\partial x \partial y}$ is the second partial derivative of f obtained by first partial differentiating with respect to y and then with respect to x.

  In particular, the **Hessian** is the collection of all second-order partial derivatives. If $f(x, y)$ is a twice differentiable function, then:

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x} \tag{3.21}$$

  and the corresponding (symmetric) Hessian matrix is the following:

$$\boldsymbol{H} = \nabla^2_{x,y} f(x, y) \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{3.22}$$

3.2 Optimization Methods

Since machine learning algorithms are implemented on a computer, the mathematical formulations are expressed as numerical optimization methods. Training a machine learning model often boils down to finding a good set of parameters. Given an objective function, finding the best value is done using optimization algorithms. By convention, most objective functions in machine learning are intended to be minimized, that is, the best value is the minimum value. Moreover, one is interested in finding the set of parameters, $\boldsymbol{x}$ which minimizes a certain objective function, and in particular:

$$\max_{\boldsymbol{x} \in \mathbb{R}^n} (f(\boldsymbol{x})) = - \min_{\boldsymbol{x} \in \mathbb{R}^n} (-f(\boldsymbol{x})) \tag{3.23}$$

$$\operatorname*{argmax}_{\boldsymbol{x} \in \mathbb{R}^n} (f(\boldsymbol{x})) = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^n} (-f(\boldsymbol{x})) \tag{3.24}$$

Intuitively finding the best value is like finding the valleys of the objective function, and the gradients point one uphill. The idea is to move downhill (opposite to the gradient) and hope to find the deepest point. Before introducing such algorithms, some useful definition and theorems are presented:

- Let $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x}^* \in \mathbb{R}^n$ is called a **local minimum point** (resp. **strict local minimum point**) of f if there exists $\epsilon > 0$ such that:

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \text{ (resp. } f(\boldsymbol{x}^*) < f(\boldsymbol{x})), \forall \|\boldsymbol{x} - \boldsymbol{x}^*\| < \epsilon \tag{3.25}$$

- Let $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x}^* \in \mathbb{R}^n$ is called a **global minimum point** (resp. **strict global minimum point**) of f if there exists $\epsilon > 0$ such that:

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \text{ (resp. } f(\boldsymbol{x}^*) < f(\boldsymbol{x})), \forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \boldsymbol{x}^* \tag{3.26}$$

- Let $f : \mathbb{A} \in \mathbb{R}$, with $\mathbb{U} \subseteq \mathbb{R}^n$ open set. If $\boldsymbol{x}^* \in \mathbb{U}$ is a local optimum point for f and f is differentiable in $\boldsymbol{x}^*$, then:

$$\nabla f(\boldsymbol{x}^*) = 0 \tag{3.27}$$

  In general, if the above condition holds, $\boldsymbol{x}^*$ is a **stationary point**.

- Let $f : \mathbb{A} \in \mathbb{R}$, with $\mathbb{U} \subseteq \mathbb{R}^n$ open set. If $\boldsymbol{x}^* \in \mathbb{U}$ is a local optimum point for f and f is twice differentiable around $\boldsymbol{x}^*$, then:

$$\nabla f(\boldsymbol{x}^*) = 0 \text{ and } \nabla^2 f(\boldsymbol{x}^*) \text{ is positive semidefinite} \tag{3.28}$$

3.2.1 Iterative Methods

Given an initial parameter vector $\boldsymbol{x}_0 \in \mathbb{R}^n$, it is possible to approximate a solution to a given optimization problem by applying **iterative methods**. In particular, by using these methods, one can compute $\boldsymbol{x}_{k+1}$ as follows:

$$\boldsymbol{x}_{k+1} = g(\boldsymbol{x}_k) \tag{3.29}$$

until convergence. In this case g is an arbitrary function. Using these methods, $\boldsymbol{x}_k \to \boldsymbol{x}^*$ for $k \to \infty$, where $\boldsymbol{x}^*$ is a stationary point (i.e. a local minimum). Since one cannot compute an infinite number of terms (there exists a truncation error), a **stopping criterion** is used. For example:

- **Absolute criterion**, which is based on first order condition, $\|\nabla f(\boldsymbol{x}_k)\| < \tau_A$, where τ_A is the chosen tolerance.
- **Absolute criterion** on succeeding values, $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \tau_{AP}$, where τ_{AP} is the chosen tolerance.
- By choosing a number N of iterations.

In order to evaluate how fast the algorithm approximates the solution and to compare two or more different algorithms, the **convergence speed** is used. In particular, let $\boldsymbol{P}x_k$ be a sequence converging to $\boldsymbol{x}^*$:

- The algorithm is **Q-linear** if it exists $r \in (0, 1)$ such that:

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} \leq r, \ \forall k > k^* \tag{3.30}$$

  Hence, the distance to the solution decreases at each iteration of a constant factor.

- The algorithm is **Q-quadratic** if it exists $M > 0$ such that:

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2} \leq M, \ \forall k > k^* \tag{3.31}$$

3.2.2 Descent Methods

Given an initial parameter vector $\boldsymbol{x}_0 \in \mathbb{R}^n$, it is possible to approximate a solution to a given optimization problem by applying **descent methods**, which are a subset of iterative methods. In particular, by using these methods, one can compute $\boldsymbol{x}_{k+1}$ as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{3.32}$$

where $\boldsymbol{p}_k$ is the **descent direction** and α_k is a positive parameter called **step size**, which measures the step to take along the direction $\boldsymbol{p}_k$. In particular, $\boldsymbol{p}_k$ is a **descent direction** if there exists $\alpha_k > 0$ such that:

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f(\boldsymbol{x}_k) \tag{3.33}$$

In this case one has that:

$$\begin{cases} \boldsymbol{p}_k^T f(\boldsymbol{x}_k) < 0 & \text{if } \nabla f(\boldsymbol{x}_k) \neq 0 \\ \boldsymbol{p}_k = 0 & \text{if } \nabla f(\boldsymbol{x}_k) = 0 \end{cases} \tag{3.34}$$

The choice of $\boldsymbol{p}_k$ corresponds to different descent methods, while the step size is usually chosen by running a line search algorithm (i.e. the suitable value for α_k is searched along the direction $\boldsymbol{p}_k$), where an iterative procedure decreases the value of the step size until suitable conditions for the convergence of the method are satisfied.

- The **gradient descent method** set $\boldsymbol{p}_k = -\nabla f(\boldsymbol{x})$, which satisfy the above-mentioned conditions (i.e. is a descent direction), so that:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}) \tag{3.35}$$

  The convergence speed of such method is Q-linear. Moreover, the convergence of such method may be very slow if the curvature of the optimization surface is such that there are regions that are poorly scaled.

- The **gradient descent with momentum** method improves the convergence property of the gradient descent method by memorizing and utilizing, at each step, information of the previous iteration. In particular:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}) + \beta \Delta \boldsymbol{x}_k \tag{3.36}$$

  where $\beta \in [0, 1]$ and $\Delta \boldsymbol{x}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ is the update obtained at iteration k. This update smooths the gradient updates and, thus, helps to avoid changing of direction.

- Given N data points (i.e. samples), the **stochastic gradient descent** method approximates the true gradient by considering a number $M < N$ of data points. In particular, in machine learning one often consideres objective functions that are the sum of the losses L_n incurred by each example n:

$$L(\theta) = \sum_{n=1}^{N} L_n(\theta) \tag{3.37}$$

The gradient of such loss function is then computed as follows:

$$\nabla L(\theta) = \sum_{n=1}^{N} \nabla(L_n(\theta)) \tag{3.38}$$

Since computing the gradient of such loss function is quite expensive, by using stochastic gradient descent it is possible to select a **mini-batch** of $M < N$ samples such that:

$$\nabla L(\theta) = \sum_{n=1}^{M} \nabla(L_n(\theta)) \tag{3.39}$$

Stochastic gradient descent converges at the same result obtained by using classical gradient descent.

- The **Newton method** set $\boldsymbol{p}_k = -\mathrm{H}_f^{-1}(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)$, where H_f is a positive definite matrix, so that:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \mathrm{H}_f^{-1}(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k) \tag{3.40}$$

In particular, at each iteration $\boldsymbol{p}_k$ is computed as the solution of the following linear system:

$$\mathrm{H}_f(\boldsymbol{x}_k)\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k) \tag{3.41}$$

The convergence speed of such method is Q-quadratic but this method is computationally more expensive than the gradient descent one.

3.3 Convex Optimization

Another useful class of optimization problems, where one can guarantee global optimality, is represented by the class of convex optimization problems. In particular:

- A set $\mathcal{C}$ is a **convex set** if for any $x, y \in \mathcal{C}$ and for any scalar θ with $0 \leq \theta \leq 1$, one has:

$$\theta x + (1 - \theta)y \in \mathcal{C} \tag{3.42}$$

From a geometrical point of view, this means that given the points x and y, every other point of the segment connecting x and y is in $\mathcal{C}$.

- Let function $f : \mathbb{R}^D \to \mathbb{R}$ be a function whose domain is a convex set. The function f is a **convex function** if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leq \theta \leq 1$, one has:

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}) \tag{3.43}$$

  If instead the left-hand side of such equation is strictly less than the right-hand side, then the function is said to be **strictly convex**.

- If $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is a convex function, then all the local minima are global minima. Moreover, if $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is a strictly convex function, then f has only one local minimum which is also a global minimum.

- If a function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is differentiable, one can specify convexity in terms of its gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. In particular, a function f is convex if and only if for any two points $\boldsymbol{x}, \boldsymbol{y}$ it holds that:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) \tag{3.44}$$

- If a function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, i.e. the Hessian exists for all values in the domain of f, the the function is convex if and only if $\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ is positive semi-definite.

The global minimum of such functions can be obtained by applying the afore-mentioned optimization methods. An example of a convex function is the one related to the linear least-squares problem ($f = \|Ax = b\|_2^2$).

Chapter 4

Probability and Statistics

4.1 Probability

Probability concerns the study of uncertainty. Probability can be thought of as the fraction of times an event occurs, or as a degree of belief about an event. Then, one could be interested in using such probability to measure the chance of something occurring in an experiment. In particular, in machine learning there are two main interpretations of probability:

- The **Bayesian probability**, which is used to quantify the degree of uncertainty that the user has about an event.
- The **frequentist probability**, which considers the frequency of events of interest with respect to the total number of events occurred.

In particular:

- A **random experiment** is an experiment whose outcome is determined by chance.
- The **sample space** Ω is the set of all possible results of a random experiment (e.g. if one rolls a dice, then $\Omega = \{1, 2, 3, 4, 5, 6\}$).
- An **event** A is a collection of results and it is a subset of the sample space (e.g. if one rolls a dice, then the possible events are $A = \{1\}$ or $A = \{2\}$ or $\dots$ or $A = \{1, 1\}$ or $A = \{1, 2\}$ or $\dots$).
- The **event space** $\mathcal{A}$ is the set of all the possible events. It usually corresponds to the power set of Ω.
- The **probability** of an event $A \in \mathcal{A}$ is a function:

$$P : \mathcal{A} \to [0, 1] \in \mathbb{R} \tag{4.1}$$

  which associates each event A to a number called probability of A, which measures the probability or degree of belief that the event will occur. In particular:

– Each probability function P satisfies:
  * $P(A) \geq 0$.
  * $P(\Omega) = 1$.
  * If $A_1, \ldots, A_n$ are disjoint events (i.e. $A_1 \cap \cdots \cap A_n$), then:

$$P(A_1 \cup \cdots \cup A_n) = \sum_{i=1}^{n} P(A_i) \tag{4.2}$$

– The **conditional probability** of an event B given the event A is defined as:

$$P(B|A) = \frac{P(A \cup B)}{P(A)} \tag{4.3}$$

For any fixed event A, with $P(A) > 0$:
  * $P(B|A) \geq 0$, $\forall B \subset \Omega$.
  * $P(\Omega|A) = 1$.
  * If $B_1, \ldots, B_n$ are disjoint events (i.e. $B_1 \cap \cdots \cap B_n$), then:

$$P\left(\bigcup_{i=1}^{n} B_i|A\right) = \sum_{i=1}^{n} P(B_i|A) \tag{4.4}$$

  * $\forall A_i, \ldots, A_n$ events:

$$P(A_1 \cap \cdots \cap A_n) = P(A_1) \cdot P(A_2|A_1) \cdot \cdots \cdot P(A_n|A_1, \ldots, A_{n-1}) \tag{4.5}$$

– Two events A and B are **independent** if and only if:

$$P(A \cap B) = P(A) \cdot P(B) \tag{4.6}$$

– Given $A, B \in \Omega$ disjoint events, with $A \cup B = \Omega$, The **Bayes theorem** states that:

$$\underbrace{P(B|A)}_{\text{posterior}} = \frac{\overbrace{P(A|B)}^{\text{likelihood}} \cdot \overbrace{P(B)}^{\text{prior}}}{\underbrace{P(A)}_{\text{evidence}}} \tag{4.7}$$

This theorem can be extended to multiple events $B_1, \ldots, B_n \in \Omega$ pairwise disjoint (i.e. $B_i \cap B_j = \emptyset$, $\forall i \neq j$) and exhaustive (i.e. $B_1 \cup \cdots \cup B_n = \Omega$):

$$P(B_i|A) = \frac{P(A|B_i) \cdot P(B_i)}{\sum_{i=j}^{n} P(A|B_j) \cdot P(B_j)} \tag{4.8}$$

4.1.1 Random Variables

A **random variable** is a function $X : \Omega \to \mathbb{R}$ which associates each outcome $\omega \in \Omega$ to a number $x \in \mathbb{R}$. In particular, the set of all the possible values of a random variable X is called **target space** or **support** of X (S_X):

- If S_X is finite or numerable (i.e. if $\#(S_X) < \infty$ or $\#(S_X) = \#(\mathbb{Z})$, where $\#$ is the cardinality function), then the random variable X is called **discrete**.
  - Each discrete random variable has a function associated to itself, called **probability mass function** (PMF), which describes the probability mapping between the event and the probability of outcome of the random variable:

    $$p_X : S_X \to [0, 1] \in \mathbb{R} \tag{4.9}$$

    such that:

    $$p_X(x) = P(X = x),\ x \in S_X \tag{4.10}$$

    and:

    $$\sum_{x \in S_X} p_X(x) = 1 \tag{4.11}$$

  - The **expectation of a discrete random variable** is defined as:

    $$\mu = \mathbb{E}[X] = \sum_{x \in S_X} x p_X(x) \tag{4.12}$$

    The expectation of a function $g(X)$ is defined as:

    $$\mathbb{E}[g(X)] = \sum_{x \in S_X} g(x) p_X(x) \tag{4.13}$$

  - The **variance of a discrete random variable** is defined as:

    $$\sigma^2 = \mathbb{E}[(X - \mu)^2] = \sum_{x \in S_X} (x - \mu)^2 p_X(x) = \mathbb{E}[X^2] - \mathbb{E}^2[X] \tag{4.14}$$

    The quantity σ is called **standard deviation of the discrete random variable**.
  - Some examples of probability mass functions are the **discrete uniform distribution**:

    $$p_X(x) = \frac{1}{n} \tag{4.15}$$

    where n is the dimension of the support, and the **Poisson distribution**:

$$p_X(x) = e^{-\lambda}\frac{\lambda^x}{x!} \tag{4.16}$$

which has $\mu = \lambda$ and $\sigma^2 = \lambda$ and it is especially used to describe the distribution of a random variable which counts the number of events happening in a unit of time.

- If S_X is not finite and numerable (i.e. if $\#(S_X) = \infty$ or $\#(S_X) = \#(\mathbb{R})$, where $\#$ is the cardinality function), then the random variable X is called **continuous**.
  - Each continuous random variable has a function associated to itself, called **probability density function** (PDF), $p(x)$, satisfying:

$$P(a \leq X \leq b) = \int_a^b p_X(x)dx, \ \forall [a,b] \in S_X \tag{4.17}$$

and

$$\int_a^b p_X(x)dx = 1 \tag{4.18}$$

  - The **expectation of a continuous random variable** is defined as:

$$\mu = \int_{S_X} xp(x)dx \tag{4.19}$$

  - The **variance of a continuous random variable** is defined as:

$$\sigma^2 = \int_{S_X} (x-\mu)^2 p(x)dx \tag{4.20}$$

The quantity σ is called **standard deviation of the continuous random variable**.

  - Some examples of probability density functions are the **normal distribution** or **Gaussian distribution**:

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{4.21}$$

with expectation μ and standard deviation σ and the **exponential distribution**:

$$p_X(x) = \lambda e^{-\lambda x} \tag{4.22}$$

which has $\mu = \frac{1}{\lambda}$ and $\sigma = \frac{1}{\lambda}$.

Moreover, considering two or more random variables, one can state the following (as described in case of events):

- The target space of each of the **joint probability** is the Cartesian product of the target spaces of each of the random variables. In particular, considering two random variables X and Y, the joint probability is defined as:

$$P(X = x_i, Y = y_i) = P(X = x_i \cap Y = y_i) = p(x_i, y_i) = \frac{n_{ij}}{N} \tag{4.23}$$

where n_{ij} is the number of events with state x_i and y_i and N is the total number of events.

- The **marginal probability** that X takes the value x irrespective of the value of the random variable Y is written as $p(x)$. In particular, the notation $X \sim p(x)$ is used to denote that the random variable X is distributed according to $p(x)$.

- If one considers only the instances where $X = x$, then the fraction of instances for which $Y = y$ is written as $p(y|x)$. This is called **conditional probability**.

- Considering two multivariate random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, with support $\mathcal{X}$ and $\mathcal{Y}$, the **sum rule** states that:

$$p(\boldsymbol{x}) = \begin{cases} \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) & \text{if } \boldsymbol{y} \text{ is discrete} \\ \int_{\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} & \text{if } \boldsymbol{y} \text{ is continuous} \end{cases} \tag{4.24}$$

- The **product rule** relates the joint distribution to the conditional distribution and the marginal distribution via:

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \tag{4.25}$$

- The Bayes theorem states that:

$$\underbrace{p(\boldsymbol{x}|\boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{x})}^{\text{likelihood}} \overbrace{p(\boldsymbol{x})}^{\text{prior}}}{\underbrace{p(\boldsymbol{y})}_{\text{evidence}}} \tag{4.26}$$

4.2 Statistics

- Considering a multivariate random variable X, the **mean** of such random variable with states $\boldsymbol{x} \in \mathbb{R}^D$ is an average and is defined as:

$$\mathbb{E}_X[\boldsymbol{x}] = \begin{bmatrix} \mathbb{E}_{X_1}[x_1] \\ \vdots \\ \mathbb{E}_{X_D}[x_D] \end{bmatrix} \in \mathbb{R}^D \tag{4.27}$$

where:

$$\mathbb{E}_{X_d}[x_d] = \begin{cases} \sum_{x_i \in \mathcal{X}} x_i p(x_d = x_i) & \text{if } X \text{ is a discrete random variable} \\ \int_{\mathcal{X}} x_d p(x_d) dx_d & \text{if } X \text{ is a continuous random variable} \end{cases} \tag{4.28}$$

- The **covariance between two univariate random variables** $X, Y \in \mathbb{R}$ represents the notion of how dependent the random variables are to one another and it is given by:

$$\text{Cov}[x, y] = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y] \tag{4.29}$$

  Moreover, $\text{Cov}[x, x]$ is called the **variance** and it is denoted as $\mathcal{V}_X[x]$. The square root of the variance is called **standard deviation** and it is denoted as $\sigma(x)$.

- The **covariance between two multivariate random variables** X, Y with states $\boldsymbol{x} \in \mathbb{R}^D$ and $\boldsymbol{y} \in \mathbb{R}^E$ is defined as:

$$\text{Cov}[\boldsymbol{x}, \boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}^T] \in \mathbb{R}^{D \times E} \tag{4.30}$$

- The **variance** of a multivariate random variable X with states $\boldsymbol{x} \in \mathbb{R}^D$ and a mean vector $\mu \in \mathbb{R}^D$ is defined as:

$$\begin{aligned} \mathbb{V}_X[\boldsymbol{x}] &= \text{Cov}_X[\boldsymbol{x}, \boldsymbol{x}] \\ &= \mathbb{E}_X[(\boldsymbol{x} - \mu)(\boldsymbol{x} - \mu)^T] \\ &= \mathbb{E}_X[\boldsymbol{x}\boldsymbol{x}^T] - \mathbb{E}_X[\boldsymbol{x}]\mathbb{E}_X[\boldsymbol{x}^T] \\ &= \begin{bmatrix} \text{Cov}[x_1, x_1] & \text{Cov}[x_1, x_2] & \dots & \text{Cov}[x_1, x_D] \\ \text{Cov}[x_2, x_1] & \text{Cov}[x_2, x_2] & \dots & \text{Cov}[x_2, x_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[x_D, x_1] & \text{Cov}[x_D, x_2] & \dots & \text{Cov}[x_D, x_D] \end{bmatrix} \end{aligned} \tag{4.31}$$

  This $D \times D$ matrix is called the **covariance matrix** of the multivariate random variable X.

- The **correlation** between two random variables X and Y is given by:

$$\text{corr}[x, y] = \frac{\text{Cov}[x, y]}{\sqrt{\mathbb{V}[x]\mathbb{V}[y]}} \in [-1, 1] \tag{4.32}$$

  In particular, positive correlation means that when x grows, they y is also expected to grow. Negation correlation means that as x increases, the y decreases.

The previous definitions of mean and covariance are usually referred as **population mean** and **population covariance**. Almost always in machine learning, one needs to learn from empirical observations of data (i.e. on a subset of the entire population). In particular:

- The **empirical mean** vector is the arithmetic average of the observations for each considered multivariate random variable and it is defined as:

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_n \tag{4.33}$$

  where $\boldsymbol{x}_n \in \mathbb{D}$.

- The **empirical covariance** matrix is a $D \times D$ matrix defined as:

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^T \tag{4.34}$$

  where $\boldsymbol{x}_n \in \mathbb{D}$.

4.2.1 Statistical Independence

Two random variables, X, Y, are **statistically independent** if and only if:

$$P(X = x, Y = y) = p(x, y) = p(x)p(y) \tag{4.35}$$

Moreover, the same two random variables are **conditionally independent** given Z if and only if:

$$p(x, y|z) = p(x|z)p(y|z) \tag{4.36}$$

Chapter 5

Central Machine Learning Problems

There are three major components to be considered in a machine learning system: data, models and learning.

5.1 Data

Data is assumed to be tabular (usually a matrix $X \in \mathbb{R}^{N \times D}$), where each row of such table contains D features and represents a particular instance or example. The elements of such table are usually represented numerically. In a supervised learning setting, one of the D row elements represents a label, and this label is used to solve, for example, regression and classification problems.

5.2 Models

Models can be implemented either as functions or as multivariate probability distributions. In particular:

- A model represented by a function, when given a particular example (i.e. a vector of features), produces an output:

$$f : \mathbb{R}^D \to \mathbb{R} \tag{5.1}$$

  One family of such models is constituted by the family of linear predictors, $f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + \theta_0$, where $\boldsymbol{\theta}$ and θ_0 are the unknowns.

- A model represented by a probability distribution is usually used to identify the true signal, in which one is interested in, from the noise affecting the data. This requires one to have a language for quantifying the effect of noise, i.e. one often would also like to have predictors that express some sort of uncertainty. This is achieved by considering models describing the distribution of possible functions (i.e. random variables).

5.3 Learning

In order to obtain a model which is able to make meaningful inferences from unseen examples, it is necessary to train such model. In particular, for the non-probabilistic model, one can follow the principle of **empirical risk minimization**, which directly provides an optimization problem for finding good parameters. With a statistical model, the principles of **maximum likelihood estimation** and **maximum a posteriori estimation** are used to find a good set of parameters.

5.3.1 Empirical Risk Minimization

This method is used to find the optimal set of parameters which optimize a given loss function. In particular:

- Assuming one is given N examples $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding scalar labels $y_n \in \mathbb{R}$, so that one can obtain the $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ pairs, one would like to estimate a predictor $f : \mathbb{R}^D \to \mathbb{R}$ which is able to find a good parameter $\boldsymbol{\theta}^*$ such that:

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}^*) = \hat{y}_n \approx y_n, \quad \forall n = 1, \ldots, N \tag{5.2}$$

- To be able to find such set of parameters, one can define a specific loss function, $\ell(y_n, \hat{y}_n)$, which produces a non-negative number representing how much error one has made on the particular prediction. Moreover, the **empirical risk** is defined as the average loss and is given by:

$$\mathbf{R}_{\text{emp}}(f, X, \boldsymbol{y}) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_n, \hat{y}_n) \tag{5.3}$$

  A specific implementation of this method is given by the least-squares problem.

- It turns out the empirical risk minimization can lead to **overfitting**, i.e. the predictor fits too closely to the training data and does not generalize well on new data. In this case, one need to apply **regularization**. In particular, regularization is a way to compromise between accurate solution of empirical risk minimization and the size of complexity of the solution. For example, one can replace the least-squares problem

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2$$

  with the regularized problem by adding a penality term involving only $\boldsymbol{\theta}$:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2$$

  The additional term $\|\boldsymbol{\theta}\|^2$ is called **regularizer** and the parameter λ is the **regularization parameter**. The regularization parameter trades off minimizing the loss on the training set and the magnitude of the parameters $\boldsymbol{\theta}$.

5.3.2 Maximum Likelihood Estimation

The idea behind maximum likelihood estimation is to define a function of the parameters that enables one to find a model that fits the data well. For data represented by a random variable $\boldsymbol{x}$ and for a family of probability densities $p(\boldsymbol{x}|\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$ (e.g. the family of Gaussian distribution $\mathcal{N}$), the **negative log-likelihood** is given by:

$$\mathcal{L}_{\boldsymbol{x}}(\boldsymbol{\theta}) = -\log p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{5.4}$$

In particular, $p(\boldsymbol{x}|\boldsymbol{\theta})$ is the distribution which models the uncertainty of the data for a given parameter setting. Considering a supervised learning setting, where one obtains pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and labels $y_n \in \mathbb{R}$, one is interested in constructing a predictor that takes a feature vector $\boldsymbol{x}_n$ as input and produces a prediction y_n, i.e. given a vector $\boldsymbol{x}_n$ one wants the probability distribution of the label y_n. The first example that is often used is to specify that the conditional probability of the labels given the examples is a Gaussian distribution. In other words, one assumes that one can explain the observation uncertainty by independent Gaussian noise with zero mean, $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$. Assuming the linear model $\boldsymbol{x}_n^T\boldsymbol{\theta}$ is used for prediction, it is possible to specify a Gaussian likelihood for each pair $(\boldsymbol{x}_n, y_n)$:

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \mathcal{N}(y_n|\boldsymbol{x}_n^T\boldsymbol{\theta}, \sigma^2) \tag{5.5}$$

Assuming that the set of examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ are independent and identical distributed, then the whole dataset composed of $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\mathcal{Y} = \{y_1, \ldots, y_N\}$ factorizes into a product of the likelihoods of each example:

$$p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{5.6}$$

In machine learning one often considers the negative log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{5.7}$$

In order to find the best set of parameters $\boldsymbol{\theta}$ it is necessary to minimize such function. Continuing with the proposed example, the negative log-likelihood can be written as:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \\
&= -\sum_{n=1}^{N} \log \mathcal{N}(y_n|\boldsymbol{x}_n^T\boldsymbol{\theta}, \sigma^2) \\
&= -\sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta})^2}{2\sigma^2}\right) \\
&= -\sum_{n=1}^{N} \log \exp\left(-\frac{(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta})^2}{2\sigma^2}\right) - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \\
&= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta})^2 - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}}
\end{aligned}
\tag{5.8}
$$

Since σ is given, the second term is a constant and should not be considered:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta})^2 \\
&\Leftrightarrow \frac{1}{2\sigma^2}\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2
\end{aligned}
\tag{5.9}
$$

Hence, minimizing such function corresponds to solve the least-squares problem.

5.3.3 Maximum A Posteriori Estimation

It turns out that maximum likelihood estimation may suffer from overfitting, analogously to unregularized empirical risk minimization. To solve this problem, the maximum a posteriori estimation is used instead. In particular, if one has prior knowledge about the distribution of the parameters $\boldsymbol{\theta}$, one can multiply an additional term to the likelihood. To do so, the Bayes theorem is used:

$$
p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} \tag{5.10}
$$

In the case of maximum a posteriori estimation, one is interested in maximizing the posterior distribution, i.e. $p(\boldsymbol{\theta}|\boldsymbol{x})$, and since the distribution $p(\boldsymbol{x})$ does not depend on $\boldsymbol{\theta}$, one can ignore the value of the denominator and obtain:

$$
p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{5.11}
$$

Since maximizing a given function is equivalent to minimizing the negative of the function:

$$\begin{aligned}\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{x}) &\Leftrightarrow \min_{\boldsymbol{\theta}} -\log p(\boldsymbol{\theta}|\boldsymbol{x}) \\ &= \min_{\boldsymbol{\theta}} -\log p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &= \min_{\boldsymbol{\theta}} -(\log p(\boldsymbol{x}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))\end{aligned} \tag{5.12}$$

Considering the same example as before, if the parameters $\boldsymbol{\theta}$ are distributed according to the same Gaussian distribution of the examples, i.e. $\boldsymbol{\theta} \sim \mathcal{N}(0, \sigma^2)$, then:

$$\begin{aligned}f(\boldsymbol{\theta}) &= -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &= \cdots \\ &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta})^2 + \sum_{n=1}^{N}\theta_n^2 \\ &\Leftrightarrow \frac{1}{2\sigma^2}\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}\|_2^2\end{aligned}$$

Where $\|\boldsymbol{\theta}\|_2^2$ is equivalent to the regularization term introduced in the empirical risk minimization method.